HAAGA-HELIA
University of Applied Sciences

Development of Shipment Management Application Using MVC4

Md.Shohidul Haque

A1003551

Thesis

09.10.2014

HAAGA-HELIA
University of Applied Sciences

Abstract

Date of presentation

Degree programme

Author Md.Shohidul Haque A1003551	**Group or year of entry** 2010
Title of report Development of Shipment Management Application Using MVC4	**Number of report pages and attachment pages** 42+4
Teacher(s) or supervisor(s) Isonikkilä Sauli	

The client for my bachelor thesis was Incepta Pharmaceuticals Ltd, a multinational company. It is situated in Bangladesh and its main office in Dhaka, capital city of Bangladesh. Mainly it produces medicines and drugs. Incepta pharmaceuticals told me to develop a prototype of an application which will have the customer, product and delivery information to replace their old one.

I had learnt C#, MySQL, JQuery, Ajax, MVC from my school Haaga-Helia University of Applied Science. During the planning of this project to develop this application I had decided that I will use the ASP.NET MVC4 Framework.

During the development, the study emphasized on defining use cases, required elicitation process, designing and implemention.Futhermore, a short description of MVC (Model, View, and Controller) were given in the document as well.

The main goal was to develop a prototype of the application for Incepta Pharmaceuticals Ltd. Moreover, the application has just customer, product and delivery pages. Furthermore, if the company wants to add more pages in the application they can follow this document.

Keywords
ASP.NET MVC,User Requirement,ViewBang,Html Helper

Contents

1 Introduction

1.1 Background of the thesis

Incepta Pharmaceuticals Ltd. Company is the client of the product which is a leading pharmaceutical company in Bangladesh which is established in 1999.The Company has a very big manufacturing facility located at Savar, 35 kilometres away from the centre of the capital city Dhaka. Now they export their products in 47 countries of the world through their distribution partner including here in Finland.

They performed their various daily activities using an application which was not of Microsoft product. Now they are interested to use the Microsoft web services and their soft wares for their business purposes .They think that Microsoft web services and their soft wares are more secured than others.

1.2 Objective of the Thesis

The main objective of this thesis was to develop a prototype of the application using ASP.NET MVC Framework which will be used by the employees of Incepta Pharmaceuticals Ltd. Company.

1.3 Scope of the project

At present they have application which was built by PHP language.PHP is an open source language though some of the tools like Intellij or PHP storm is needed to buy. However, the company wants to develop a large scale application so that can they can maintain their customers' products' and all other information using the same application. In addition to this, they interested to use the Microsoft product into their system.ASP.NET MVC is faster than PHP.Moreover ASP.NET MVC has a concept of whole application whereas

PHP does not. I had learnt ASP.NET MVC4, C#, JQuery, JavaScript from my school – Haaga-Helia University of Applied Science. That is why I decided to build up the prototype of the product for that company.

2 Technical knowledge

2.1 ASP.NET MVC Framework

I will use Microsoft ASP.NET MVC4 framework for developing this product. I have chosen ASP.NET MVC4 framework over ASP.NET Web Form because in MVC4 it is easier to handle the application by splitting into Model, View, and Controller. Moreover I have chosen this framework because I learnt this technology from my school Haaga-Helia University of Applied science.

ASP.NET MVC (Model View Controller) is a framework for building web application that applies the general Model View Controller pattern to the ASP.NET framework.

2.2 ASP.NET MVC4 Framework

ASP.NET MVC (Model View Controller) framework is a web development framework to develop the web application. In addition to this, Model contain Business logic layer, View contains user interface layer and controller contain input logic layer. Moreover, developer can easily concentrate into the individual parts of this framework to develop the application.

MVC 4 includes a better solution: ASP.NET Web API (referred to as Web API), a framework that offers the ASP.NET MVC development style which lead to writing HTTP services. This includes both modifying some ASP.NET MVC concepts to the HTTP service domain and supplying some new service-oriented features.(Galloway, Jon, Haack, Phil, and Wilson, Brad 2012,10)

MVC4 can be run on the windows operating system like-Windows XP, Vista, and win-dow8.Moreover; it can be run on the Windows server2003, server2008, and server2008R server operating system.

2.3 Model

The model classes are needed to deliver entity information between different parts of the framework. We could use two separated approaches to create model classes.

1) Create EDM(Entity Data Model) from the database
2) Create your own class.

A model can be anything from a single object instance, to a collection of instance, even to a complicated object structure. While using the EDM Model the Model transports both the real data and metadata about the model type.

2.4 View

The View is responsible for

1. The web page structure (HTML5 code)
2. Showing the model data on the page
3. For providing input controls and links for doing something on this page or going to another.

Any View can be strongly-typed View or ordinary View

A strongly typed view has information about model class data type (Passed silently via the View Bang)

A View can get information from the controller the following ways

1. As a parameter (model class objet or a model class object collection)
2. Inside ViewBang collection (Or older way: View data collection9 where it can be:

- A SelectList collection of items based on relationship between the Entity class(or tables)
- Anything else that can be needed to build the user interface

HTML code should be rendered only inside a View not in the other parts (M, C) of the MVC project

2.5 Controller

Controller is %Controller.cs class with c# or BV.NET code .It is the main actor among the MVC components. When browser send request, controller class receives the request and processes the request using model data and finally selects a view to present the result to the browser (Dangar 2010, Microsoft 2013a.)

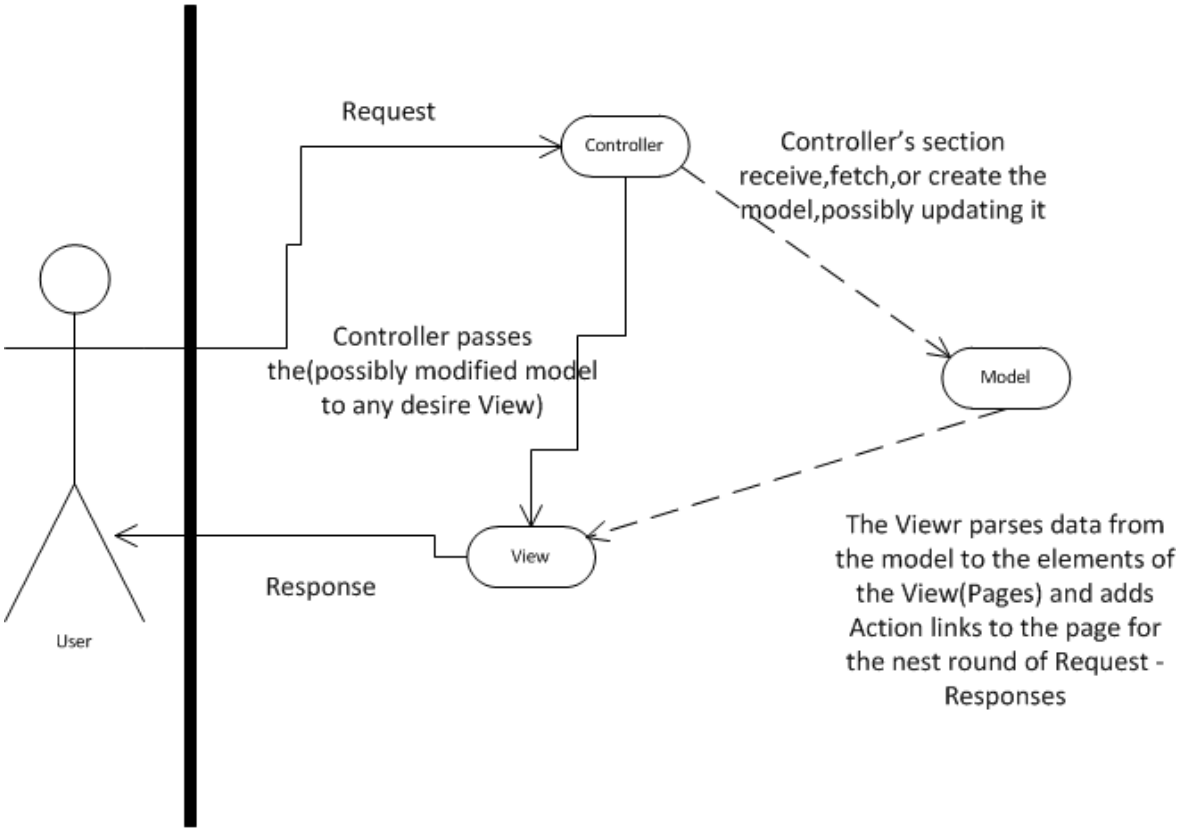


Figure1. Model View Controller structure

3 Requirement elicitation and requirement specifications

3.1 Requirement elicitation

Requirement elicitation is concerned with learning and understanding the needs of users and project sponsors with the ultimate aim of communicating these needs to the system developers. (Aybuke Aurun & Claes Wohlin 2006)

Davis and Olson (1985) have identified four strategies for determining user information requirements

1. Asking
2. Deriving from an existing information system
3. Synthesizing from the characteristics of the utilizing system
4. Discovering from experimentation with an evolving information system.

(Mohapatra, P.K.J 2010)

During my requirement elicitation process I had used combination of two techniques.

- Interview
- Brain storming

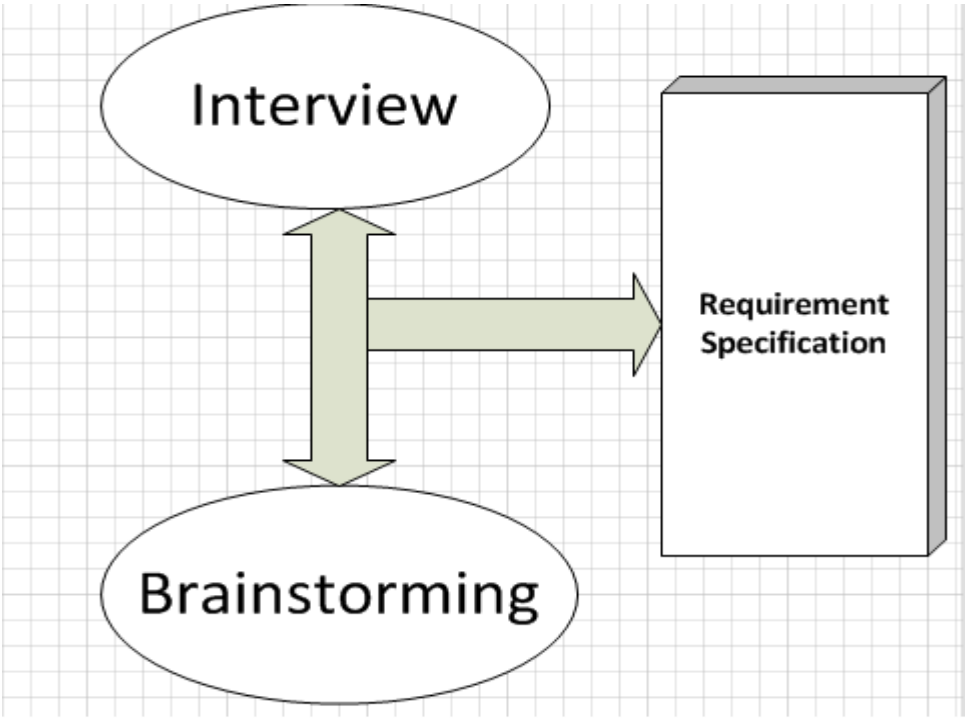


Figure2.Requirement elicitation technique

3.1.1 Interview

While questionnaire surveys are relatively easy to organize they do have certain limitations, particularly in the lack of flexibility of response. Interviews are more suitable for questions that require probing to obtain adequate information. (Nicholas Walliman 2011, 97)

There are three types interview:

- Structured interview
- Unstructured interview
- Semi-structure interview

During the collection of requirement specification, I asked face to face questions to the person who is responsible for the IT department. Questions are given in appendix 1.

3.1.2 Brain storming

Brain storming is a tool which helps to get the new idea in an environment free of criticism. During the brain discussion we finalized the requirement specifications for the application which will be developed for Incepta Pharmaceutical ltd.

4 Requirement specification

User stories are short, simple description of product described from the view of the person who wants to use the product, usually a user or customer of the system. It gives us quick idea of the software's requirement specification. The end user of the product "Development of shipment management application using MVC" is the employees of the Incepta Pharmaceuticals limited Company. So the user stories will be described from the company's personnel's point of view.

The user stories or product backlogs are as follows:

As a company's personnel I want to

- Add customer's information
- Update customers information
- See the details of customers' information
- Delete the customer information
- Add product details in the system
- Update product information in the system
- Display product information
- Delete product information
- Add delivery information
- Update delivery information
- See the delivery information
- Delete delivery information

5 Use case diagram

Conceptual use case Model is given below. It is normative model that have been implemented to show the interaction between end users and the system. Use cases would be modified development progress further.

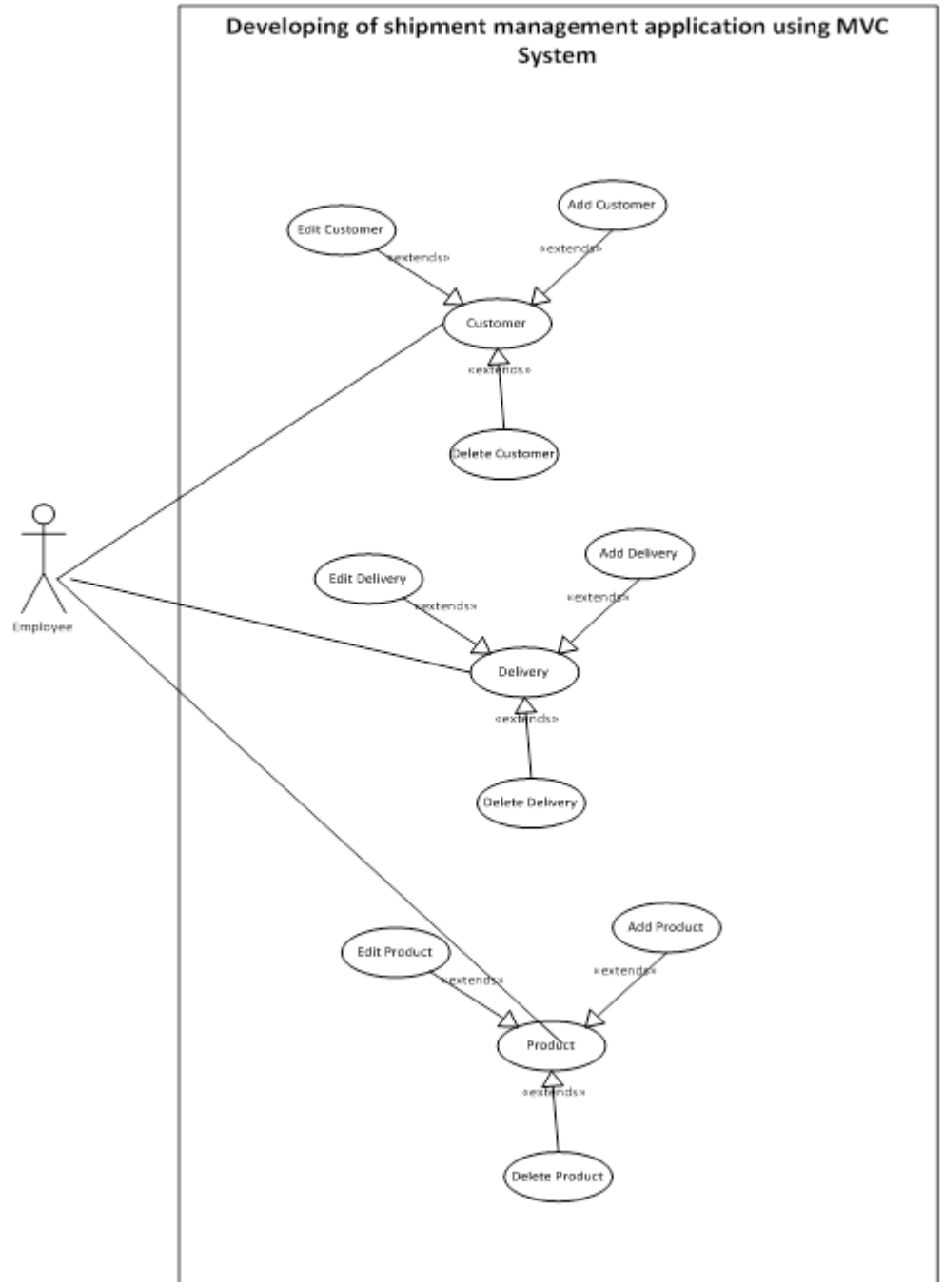


Figure3.Use Case Diagram

5.1 Description of the system's use case diagram

5.1.1 Add Customer

Add Customer use case serves to input the customer information to the customer database. The Actor fills in the information like customer id, customer name, phone, email, street address, post code and city.

5.1.2 Edit Customer

The actor can change the customer information in the system if the customer informs the actor. Customers can inform the actor to update their information by phone or by email. The actor is able to edit customers' information about change of customeid, customername.

5.1.3 Delete Customer

The actor deletes the customer information from the system so that it no longer exists in that database.

5.1.4 Actor

Here the actor will be the employees of the Incepta Pharmaceuticals Ltd. Company. He or she is the responsible person to receive customer order by phone or e-mail and enter them into the system. Moreover, he or she also can update or delete the customers' information from the system.

5.1.5 Add product

Add Product use case serves to input the product information to the product database. The Actor fills in the information like product id, product name, weight, price, description into the system.

5.1.6 Edit Product

The actor updates product information into the system. After delivery the product, actor checks the inventory situation and then actor edit the product information in the system.

5.1.7 Delete product

The actor deletes product information from the system so that it no longer exists.

5.1.8 Add Delivery

The actor adds the delivery information into the system like delivery id and delivery date and time. Delivery information is added when customer calls the actor by phone or e-mail.

5.1.9 Edit Delivery

The actor updates the delivery information in the system if the delivery is done or postponed. The actor is able to edit deliveries information about change of deliveryId, deliveryDateAndTime.

5.1.10 Delete Delivery

The actor deletes the delivery information from the delivery database so that it is no longer exists.

6 Use Case Description

6.1 Add New Customer

Use Case	Add New Customer
Actor	Employee
Pre-Condition and trigger	Customer will contact with employee calling, mailing or over phone
Goal	Add a new customer, store new customer information in the system
1.	Employee enters customer information
2.	Employee requests the system to save the provided customer information
3.	Actor write the detail information of customers
4.	Fulfil the requirement

6.2 Edit Customer

Use Case	Edit Customer
Actor	Employee
Pre-Condition and trigger	Employee previously retrieved customer information
Goal	Customer information is updated in the system
1.	Employee provides the system with updated customer information

2.	Employee requests the saving of updated information in the system

6.3 Delete Customer

Use Case	Delete Customer
Actor	Employee
Pre-Condition	Delete Customer Information
Goal	Customer Information will be deleted from the system
1.	Select customer to be deleted
2.	Delete customer

6.4 Add product

Use Case	Add Product
Actor	Employee
Pre-Condition and trigger	Product will be in the list of product of that company
Goal	To add a new product in the list of product, store new product information in the system
1.	Employee enters product information
2.	Employee requests the system to save the provided product information
3.	System stores product information, only in case of successful validation

4.	Employee write the detail information of product
5.	Fulfil the requirement

6.5 Edit Product

Use Case	Update Product
Actor	Employee
Pre-Condition and trigger	Company has product and product information has been successfully retrieved
Goal	New product is added. Product information is updated in the system.
1.	New request to edit product information
2.	Employee enters new product information data
3.	Employee submits the request
4.	New request to update customer site data
5.	System validates entered information

6.6 Delete Product

Use Case	Delete Product
Actor	Employee
Pre-Condition	Existing product, net yet informed by inventory company

Goal	Delete product information
1.	Select current product
2.	Select product to be deleted
3.	Delete product

6.7 Add Delivery

Use Case	Add Delivery
Actor	Employee
Pre-Condition and trigger	Existing Customer
Goal	Make a complete delivery
1.	Select customer for delivery
2.	Set customer for delivery
3.	Set contact person for delivery
4.	Send confirmation of delivery to customer

6.8 Update Delivery

Use Case	Update Delivery
Actor	Employee
Pre-Condition and trigger	Existing Customer
Goal	Make a complete delivery
1.	Select current delivery
2.	Select Delivery to be updated
3.	Update delivery as necessary
4.	Save delivery and generate new confirmation

6.9 Delete Delivery

Use Case	Delete Delivery
Actor	Employee
Pre-Condition and trigger	Existing Delivery, delivery not yet made
Goal	Delete delivery information
1.	Select current delivery
2.	Select delivery to be deleted
3.	Delete delivery
4.	Get confirmation of delete and send to customer

7 System Architecture Design

Architectural design is a creative process where we try to establish a system organization that satisfies the functional and non-functional system requirements. (Ian Somerville 2007, 245) The product will be used by the employee of the company. The system architecture of the product is as follows

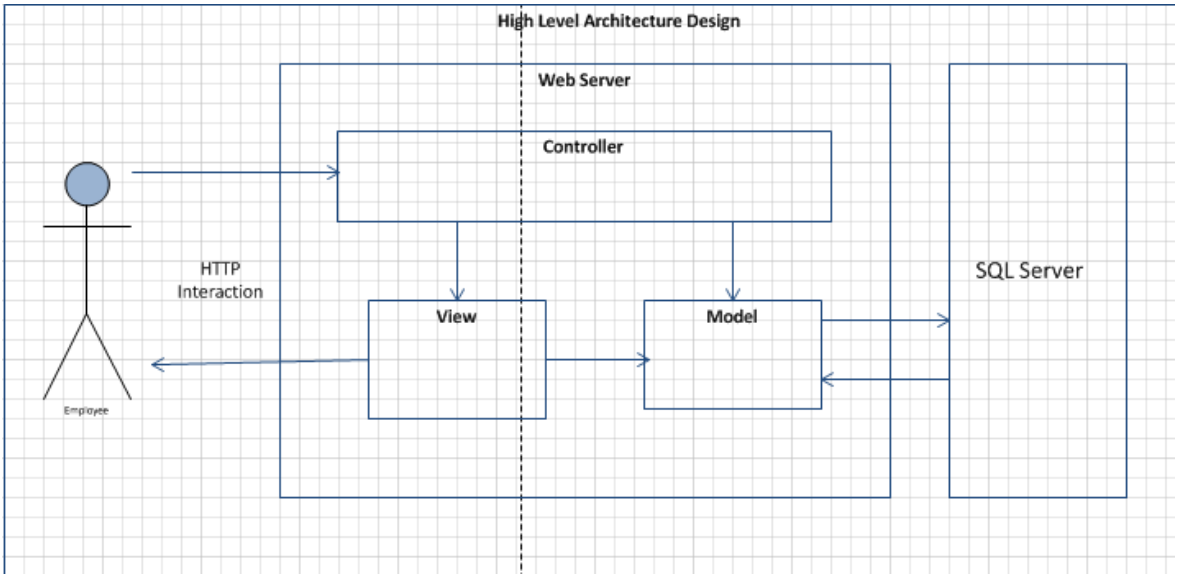


Figure4.High level architecture design

8 Conceptual Class Diagram

The conceptual level class diagram below shows the real life entities and the relationship between them. It is an information system scope class diagram that accounts for the information that is to be stored in the system. Moreover, it will be changed during the development process depending on the requirement of the company.

I created this diagram using Microsoft Visio and then I edited this diagram using the Paint software.

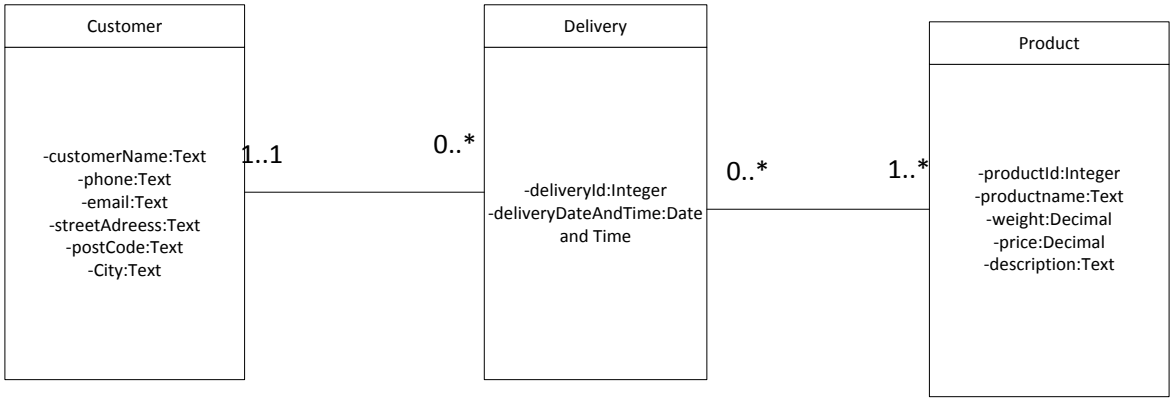


Figure5.Class Diagram

9 Classes

In this project I am using the ASP.NET MVC4 (Model View Controller) Framework .Here the classes are the classes which contain all application logic (business logic, validation logic and data access logic) except pure view and controller logic. (W3school.com 2015)

Development of Shipment Management Application system will have the following classes.

9.1 Customer

Customer class modeling a real life business customer of company.

Attributes:

cutomerId	(Integer)
cutomerName	(String)
phone	(String)
email	(String)
streetAddress	(String)
postcode	(String)
city	(String)

Relationship:

Delivery

Operation:

get & set

9.2 Product

Customer class modeling a real life business customer of company.

Attributes:

cutomerId	(Integer)
cutomerName	(String)
phone	(String)
email	(String)
streetAddress	(String)

postcode	(String)
city	(String)

Relationship:

Delivery

Operation:

get & set

9.3 Delivery

An order can be delivered with one or more deliveries.

Attributes:

Customer	(Object)
Product	(Object)
deliveryId	(Integer)
deliveryDateAndTime	(DATETIME)

Relationships:

Customer

Product

Operation:

get & set

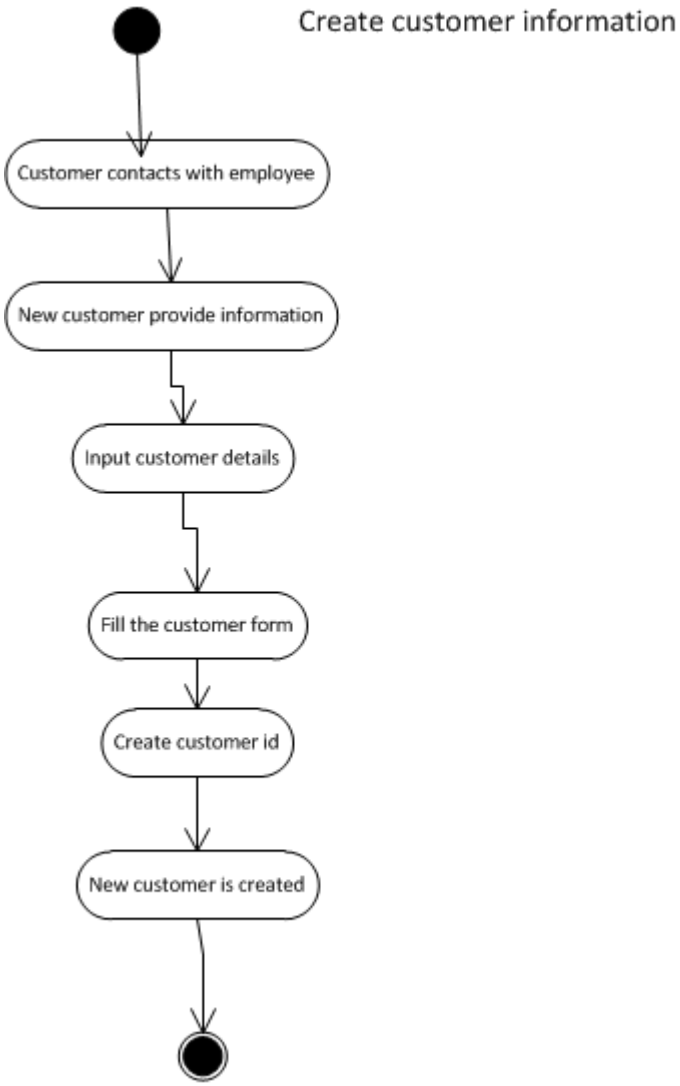
10 Activity Diagram

An activity diagram is a state transition diagram that consists of state and transition between states.it is a graphical representation of a scenario and its constituent activities. A state captures a snapshot of the system during execution. A transition capture the transition of the system from one state to another brought by performing an activity. (Ghinwa Jalloul 204, 25)

I created this diagram using Microsoft Visio and then I edited this diagram using the Paint software. In contrary, these diagrams are self-directories diagram.

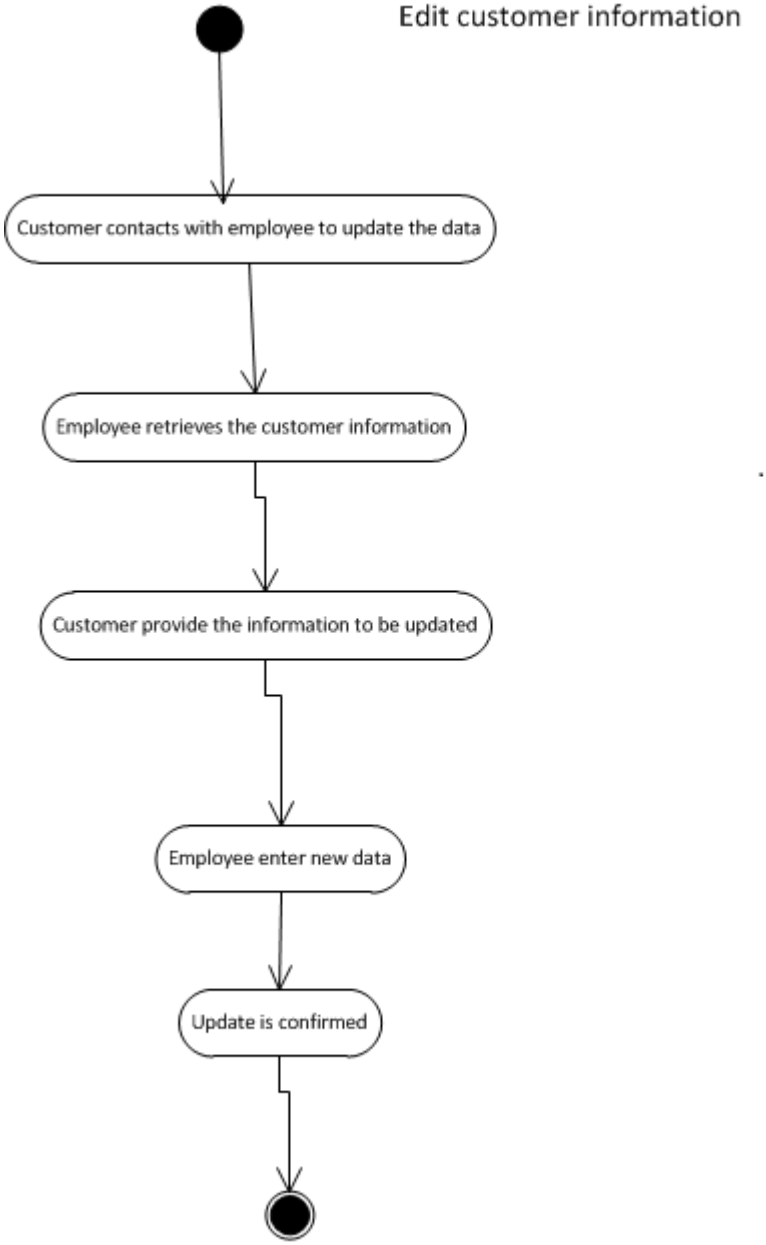
10.1 Add New Customer

This activity diagram speaks itself. This diagram, however, shows how a customer can contact with the employee through e-mail. Phone to add their information into the system.



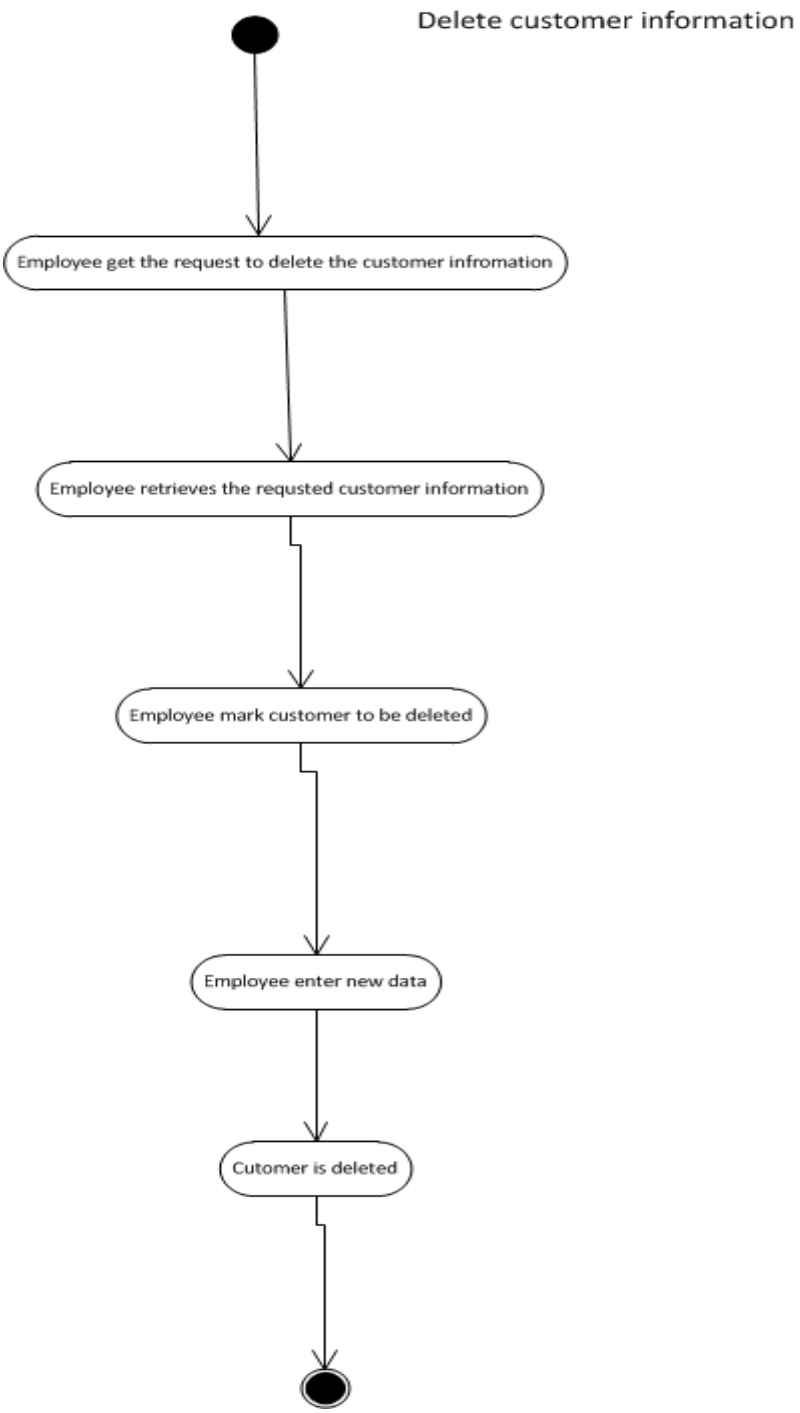
10.2 Edit Customer

This diagram shows the activities that how the customers' information are edited for the system.



10.3 Delete Customer

This is how the delete activities are done for the system



11 ER (Entity Relationship Diagram)

Conceptual Modeling is the representation of data requirements of an organization that is to be supported by the database. In database design, a conceptual data model is often presented as an Entity Relationship(ER) diagram.

In the ER diagram,

Real-World objects are represented as entities and Entities are described using attributes.

IN my project Customer, Product and Delivery are the entities and attributes are in their respective diagram. In contrary, this is not the final or static one .It will be changed during the development process depending on the requirement of the company.

I created this diagram using Microsoft Visio and then I edited this diagram using the Paint software.

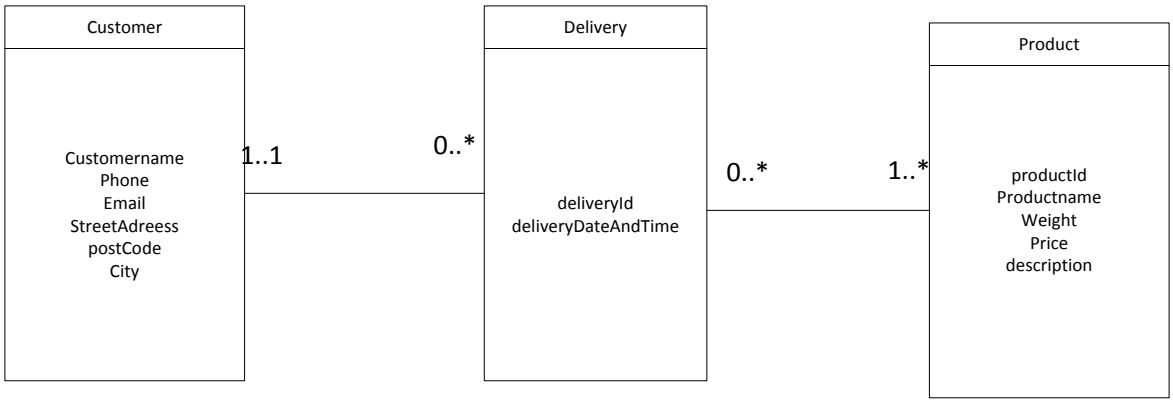


Figure 6.Entity relationship diagram

Product delivery belongs to one and exactly one customer whereas customer has zero or more delivery. Likewise, Product has zero or more delivery whereas delivery belongs to one or more product.

11.1 Data Dictionary

Table2.Data dictionary

Entity Name	Description	Aliases	Occurrence	Attributes
Customer	The customer that order product by mailing ,phone the employee	Client	Customer may be anyone that might order from company	
Product	Product that will be sold to the customers	Goods	Employees add new product constantly over the year and sometime delete the old one	
Delivery	Certain delivery contain certain amount of product	Delivery	Each delivery has one or more that product	

11.2 Database diagram

I created this diagram using the ASP.NET MVC4 Entity Framework in Microsoft Visual studio 2012 in my school Haaga-Helia University Of Applied Science. Moreover, It will be changed during the development process depending on the requirement of the company. Furthermore, I took the snapshot of this diagram to edit it using the Paint software

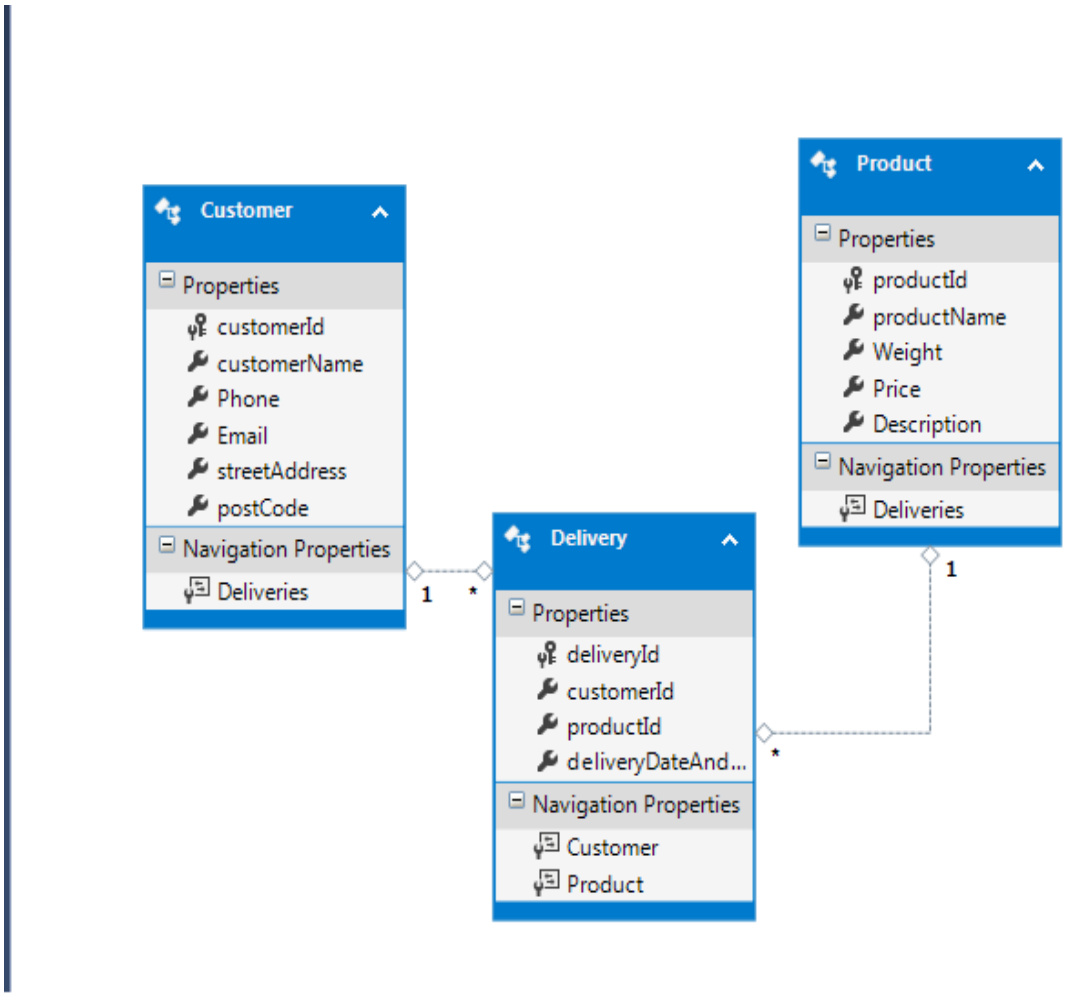


Figure 7.Database diagram

12 User Interface Diagram

User interface is the way of interaction between computer and human being. A user interface is a set of menus and commands through which a user communicate with the application. This is the most important part of the application because user always wants the most usable product.

I developed these user interfaces (UI) diagrams using the ASP.NET MVC4 Framework in Microsoft Visual studio 2012 in my school Haaga-Helia University of Applied Science. Furthermore, I took the snapshot of these diagrams to edit it using the Paint software .

12.1 Home Page

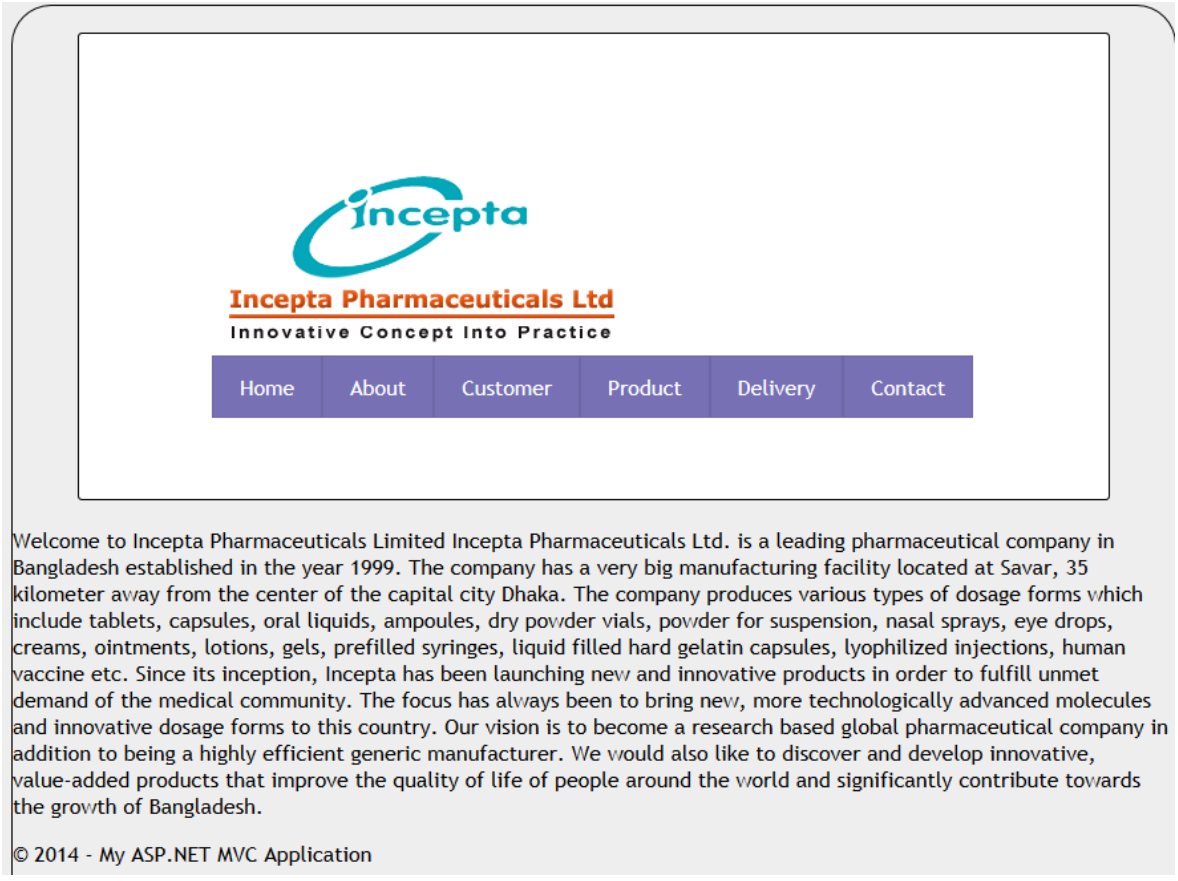


Figure8.Home page

When the application is started after the code generation the home pages is loaded first. The entire pages in the application have the same master page. The master page code can be found from appendix 2.When user will navigate the home page of the master page; the page will be seen as above figure8.It is the index.cshtml file which represent the home page for the application. Moreover this is application's default file. In the above figure we can see the logo of the company is situated above the main navigation bar and texts are displayed below the navigation bar.

12.2 Customer Page

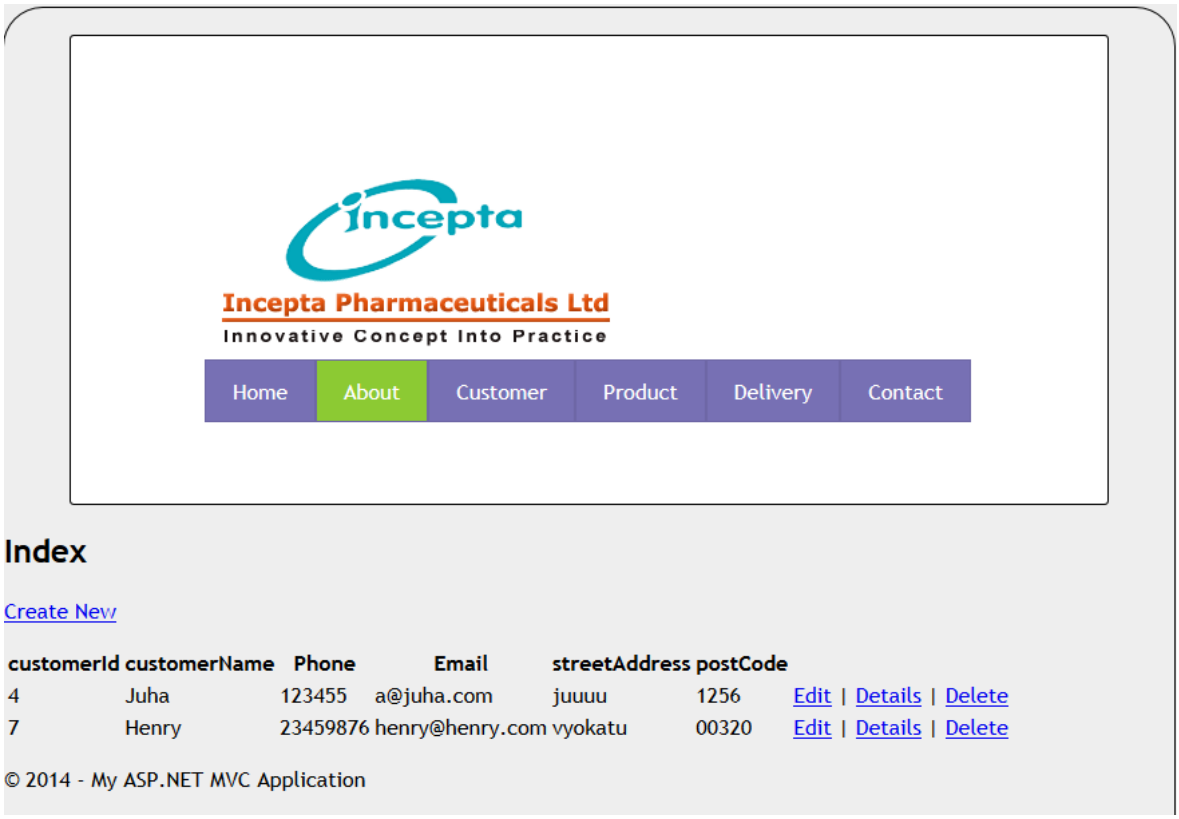


Figure9.Customer page

This is the default page for the customer. When user will navigate the customer menu in the application the above information will be seen as given in figure9. It is the index.cshtml file which represents the home page for the application which is given in appendix 3.

12.3 Product Page

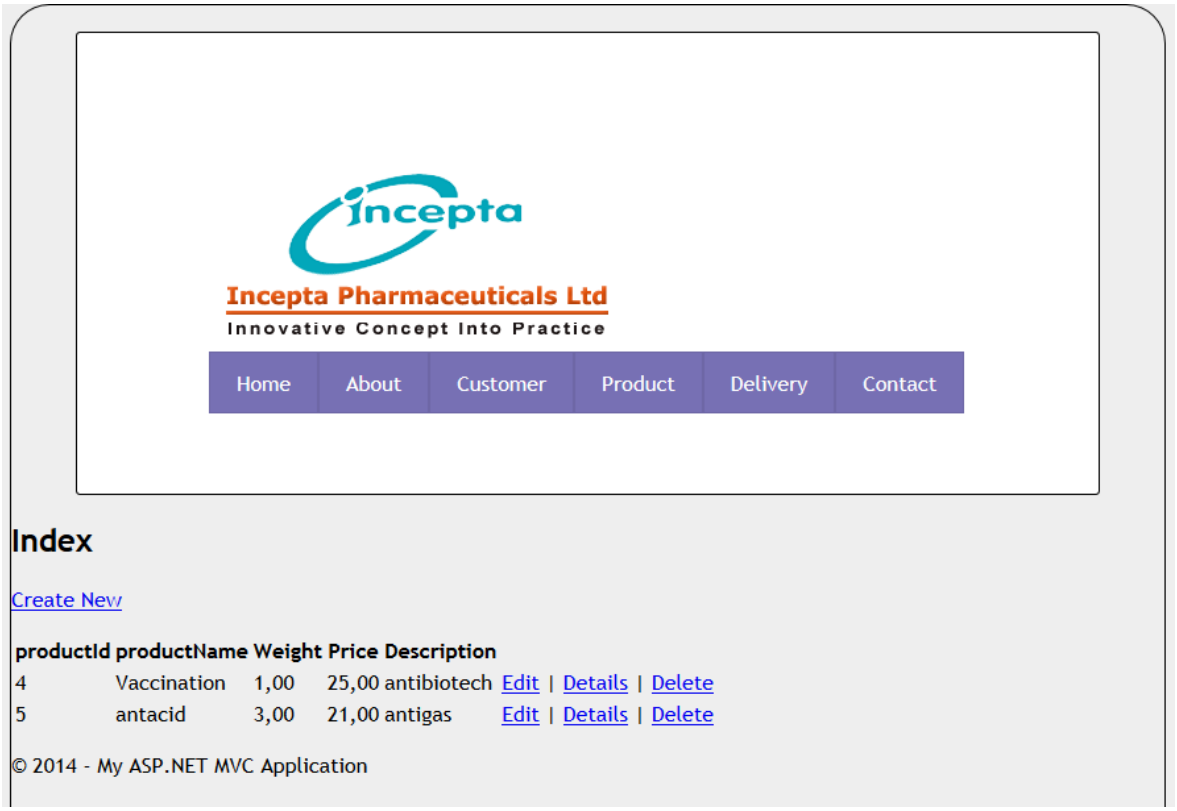


Figure10.Product page

The home page for product as seen in figure 10.This is the default page for product as well. Whenever user will navigate the product menu in the application this page will be displayed for the user. Moreover, users can edit and delete the product's information from the application as seen figure 10.

12.4 Delivery page

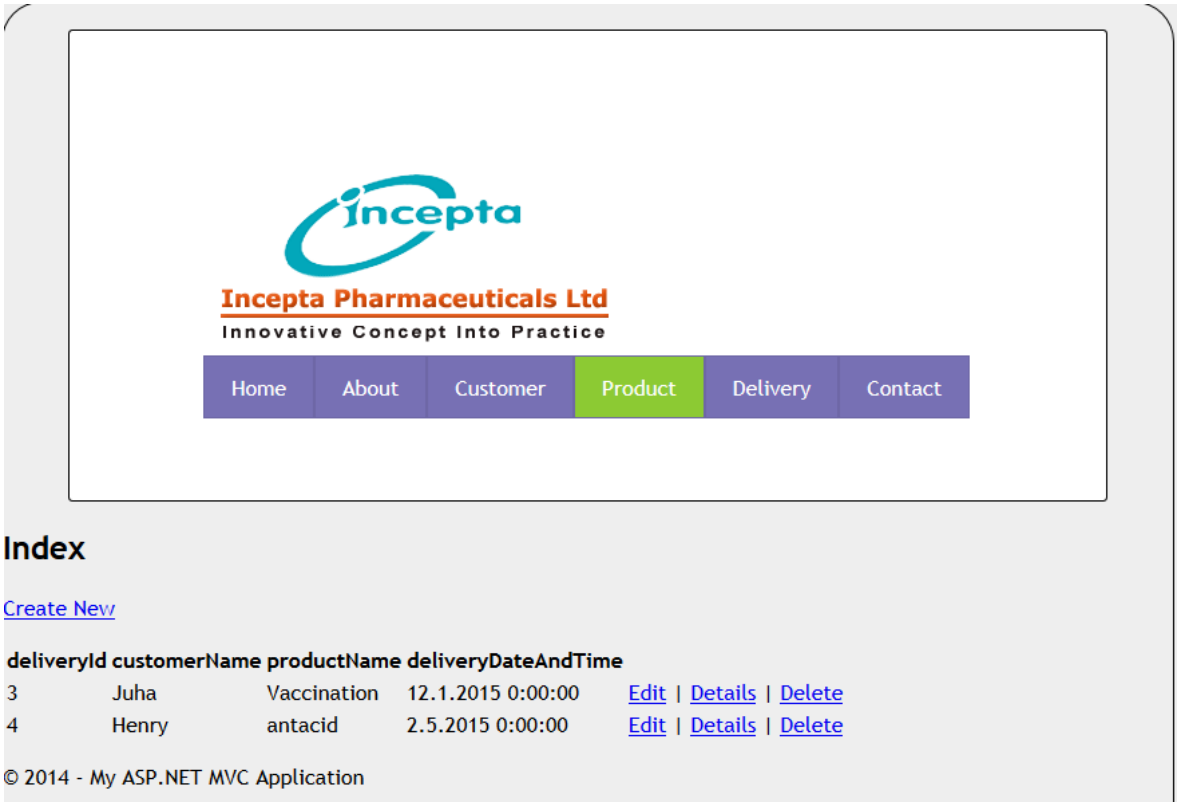


Figure11.Delivery page

This is the default page for the delivery. When user will navigate the delivery menu in the application the above information will be seen as given in figure9.

13 Implementation

This implementation will describe the code level architecture of the system. For avoiding the repetition of the same information, all parts will not be explained in this report. In contrary, the main part will be explained in this report.

13.1 Common Layout for all pages implementation

The _Layout.cshtml file which is written by c# code, represent the common layout for each page in the application. It is situated in the application of the shared folder inside the view folder. The code of the file which is given in the appendix 2, HTML helpers is used to modify output. For instance

@ViewBag.Title-the page title Developing Of Shipment Management Application Using MVC4 is inserted here.

@Styles.Render-The style sheet is inserted here.

@Render.Body ()-the page content will be rendered here.

@Html.Actionlink ()-Here Html link for Home, About, Customer, Product, and Delivery are inserted.

13.2 Home page for Customer implementation

The file index.cshtml for Customer which is given in the apendix3 represents the home page for the Customer.

This is the default file for the Customer. Here some Html helpers like @Html.DisplayFor and @HtmlDisplayNameFor are used to render the Html output.

13.3 Edit page for customer implementation

The file Edit.cshtml for customer which is given in the apendix4 helps to edit the customers' information in the database. There are some html helpers in the file which help to update the information of the customers'.

@Html.LabelFor () creates the label for that specific property for example in this file customerId,customer name,phone,email,street address and postcode will be displayed in the edit page

@Html.EditorFor () allow the developer to control over display of form elements by data type like int, Boolean, string etc.

@Html.ValidationMessageFor () is a html helper that helps to display the validation error message if model state is not valid. It helps to display a message that usually specified on top of the property in your view model.

14 Conclusion

The thesis was to develop the prototype of the application for Incepta Pharmaceuticals Ltd. I was alone to write the documentation and to develop this application as well. The application is not fully functioning but most of the functionality is working very nicely. In contrary, ASP.NET MVC4 Framework has been used to develop this prototype of the application

In some places reader might not get the clear cut idea about the business requirements, classes, and database because of changing during the development process. Sometimes I could not give enough information design and testing phases though those were in my mind. Still now I am learning and I believe I can give more concentration in future.

References

Haaga-Helia University of Applied Science 2014.ProjectPlan

http://hhmoodle.haaga-helia.fi/course/view.php? Id=326: 14 October 2014.

Inceptapharma 2014.

http://www.inceptapharma.com/index.php: 14 October 2014.

W3Schools 2014.

http://www.w3schools.com/aspnet/mvc_intro.asp: 15 October 2014.

Developing an online store for a startup apparel business

http://www.theseus.fi/bitstream/handle/10024/58478/Thesis_SayedRakibulHasan.pdf 4th march2015

John Galloway, hill Haack, Brad Wilson, K.Scott Allen 2012.Professional ASP.NET MVC4. John Wiley & Sons, 2012.

Scrum Guide

http://www.scrumguides.org/scrum-guide.html#artifacts: 12 November 2014.

Research Methods: The basic

Nicholas Walliman 2011. Research Methods. The basic.Routledge 2011.

UML by Example

Ghinwa Jalloul 2004. UML by Example: Cambridge University Press 2004.

Software Engineering

Ian Summerville 2007.Software Engineering: Harlow, England; Toronto: Addision-Wisey, 2007.

Engineering and Manging Software Requirements

Aybuke Aurun,Claes Wohlin. Engineering and Manging Software Requirements: Springer Science & Business Media 2006.

Software Engineering.

Mohapatra, P.K.J.. Software Engineering. Daryaganj, Delhi, IND: New Age International, 2010. ProQuest ebrary. Web. 1 January 2015.

Appendices

Appendix1.Interview with It responsible person of Incepta Pharmaceuticals Ltd. company.

Name: Md.Anwar Hossain Age:45 Profession: It Specialist		
1	Question	Why are you switching to the Microsoft product?
	Answers	Microsoft web service are much more secured than other services
2	Question	What information should be available in the product?
	Answers	The product should have the customer, product and delivery information
3	Question	Do you tell me whether product should be final or prototype?
	Answers	The product no need to be final product but prototype is quite ok

Appendix2.Common layout code file for all pages implementation

```
<!DOCTYPE html>
<html lang="en">
<head>
    @**@
    <meta charset="utf-8" />
    <title>@ViewBag.Title - Developing Of Shipment Management Application Using
MVC</title>
    <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
    <meta name="viewport" content="width=device-width" />
    @Styles.Render ("~/Content/styles.css")
    @Scripts.Render ("~/bundles/modernizr")
    <link href="~/Content/themes/base/jquery.ui.datepicker.css" rel="stylesheet" />
    <script src="@Url.Content ("~/Scripts/jquery.ui.datepicker.js")"
type="text/javascript"></script>
    <script src="http://code.jquery.com/jquery-1.9.1.js" type="text/javascript"></script>
    <script src="http://code.jquery.com/ui/1.10.2/jquery-ui.js"
type="text/javascript"></script>
    <script src="http://code.jquery.com/jquery-1.9.1.js" type="text/javascript"></script>
```

```
    <script type="text/javascript">
        $(function () {
            $(".datepicker").datepicker ({dateFormat: 'dd-mm-yy' });
            $(".datepicker").datepicker ();
        });
    </script>
  </head>
<body>
    <div class="example">
        <img src="~/Images/InceptaLogo.png" />
        <ul class="nav">
            <li>@Html.ActionLink("Home", "Index", "Home")</li>
            <li>@Html.ActionLink("About", "About", "Home")</li>
            <li>@Html.ActionLink("Customer", "Index", "Customer")</li>
            <li>@Html.ActionLink("Product", "Index", "Product")</li>
            <li>@Html.ActionLink("Delivery", "Index", "Delivery")</li>
            <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
        </ul>
        <div style="clear:both"></div>
    </div>
</body>
<div id="body">
    @RenderSection ("featured", required: false)
    <section class="content-wrapper main-content clear-fix">
        @RenderBody ()
    </section>
</div>
<footer>
```

```
    <div class="content-wrapper">
        <div class="float-left">
            <p>&copy; @DateTime.Now.Year - My ASP.NET MVC Application</p>
        </div>
    </div>
</footer>
@RenderSection ("scripts", required: false)
</html>
```

Appendix3.Home page code file for Customer implementation

```
@model IEnumerable<DevOfShipMangMVC.Models.Customer>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink ("Create New", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.DisplayNameFor (model => model.customerId)
        </th>
        <th>
            @Html.DisplayNameFor (model => model.customerName)
        </th>
        <th>
            @Html.DisplayNameFor (model => model. Phone)
        </th>
        <th>
            @Html.DisplayNameFor (model => model. Email)
        </th>
        <th>
            @Html.DisplayNameFor (model => model.streetAddress)
        </th>
        <th>
            @Html.DisplayNameFor (model => model. Postcode)
        </th>
        <th></th>
    </tr>
```

```
@for each (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor (model Item => item.customerId)
        </td>
        <td>
            @Html.DisplayFor (model Item => item.customerName)
        </td>
        <td>
            @Html.DisplayFor (model Item => item. Phone)
        </td>
        <td>
            @Html.DisplayFor (model Item => item. Email)
        </td>
        <td>
            @Html.DisplayFor (model Item => item.streetAddress)
        </td>
        <td>
            @Html.DisplayFor (model Item => item. Postcode)
        </td>
        <td>
            @Html.ActionLink ("Edit", "Edit", new {id=item.customerId }) |
            @Html.ActionLink ("Details", "Details", new {id=item.customerId}) |
            @Html.ActionLink ("Delete", "Delete", new {id=item.customerId})
        </td>
    </tr>
}s

</table>
```

Appendix4.Edit page code file for customer implementation

```
@model DevOfShipMangMVC.Models.Customer

@{

    ViewBag.Title = "Edit";

}

<h2>Edit</h2>

@using (Html.BeginForm ()) {

    @Html.AntiForgeryToken ()

    @Html.ValidationSummary (true)


    <fieldset>
```

```
<legend>Customer</legend>
@Html.LabelFor (model => model.customerId)
<div class="editor-field">
    @Html.EditorFor (model => model.customerId)
    @Html.ValidationMessageFor (model => model.customerId)
</div>
@Html.LabelFor (model => model.customerName)
<div class="editor-field">
    @Html.EditorFor (model => model.customerName)
    @Html.ValidationMessageFor (model => model.customerName)
</div>
<div class="editor-label">
    @Html.LabelFor (model => model. Phone)
</div>
<div class="editor-field">
    @Html.EditorFor (model => model. Phone)
    @Html.ValidationMessageFor (model => model. Phone)
</div>
<div class="editor-label">
    @Html.LabelFor (model => model. Email)
</div>
<div class="editor-field">
    @Html.EditorFor (model => model. Email)
    @Html.ValidationMessageFor (model => model. Email)
</div>

<div class="editor-label">
    @Html.LabelFor (model => model.streetAddress)
```

```
        </div>
        <div class="editor-field">
            @Html.EditorFor (model => model.streetAddress)
            @Html.ValidationMessageFor (model => model.streetAddress)
        </div>
        <div class="editor-label">
            @Html.LabelFor (model => model. Postcode)
        </div>
        <div class="editor-field">
            @Html.EditorFor (model => model. Postcode)
            @Html.ValidationMessageFor (model => model. Postcode)
        </div>
        <p>
            <input type="submit" value="Save" />
        </p>
    </fieldset>
}
<div>
    @Html.ActionLink ("Back to List", "Index")
</div>
@section Scripts {
    @Scripts.Render ("~/bundles/jqueryval")
}
```